# Assignment 6

### ****Interoperability trilemma****

*We talked before about scalability trilemma where L1 must sacrifice either decentralization, security or scalability. In a cross chain world there is also a famous interoperability trilemma.*

*Interoperability protocols can only have two of the following three properties:*

* ***Trustlessness***
* ***Extensibility***
* ***Generalizeability***

1. *Explain each aspect of the interoperability trilemma. Provide an example of a bridge protocol explaining which trade-offs on the trilemma the bridge makes.*

**[ANSWER]**

1. ***[Bonus]*** *Are there any projects that focus on solving trilemma similarly like Ethereum solves the scalability issue? If yes describe how it solves the problem.*

**[ANSWER]**

### ****Ultra light clients****

*This week we focused on how to achieve ZK interoperability between chains. How we can use Zero Knowledge Proofs in order to sync a light client faster. This is specially useful for mobile first blockchains like Celo, but is very useful for any blockchain.*

1. *Describe a specification for a light client based on zero-knowledge proofs. You should explain at least how to get the client synced up to the current state of the blockchain. Preferably go as far as explain how transaction inclusion proofs are generated too.*
2. *What is the relevance of light clients for bridge applications? How does it affect relayers?*
3. *Suppose code from* [Plumo](https://github.com/celo-org/plumo-prover) *is updated and was working in production on* [Celo](https://github.com/celo-org)*. What would be the main difficulty in porting Plumo over to Harmony?*

### ****Horizon Bridge****

*Horizon is Harmony’s bridge which allows crossing assets from Harmony to Ethereum/Binance and vice versa.*

1. *Check out* [*Horizon repository*](https://github.com/harmony-one/horizon)*. Briefly explain how the bridge process works (mention all necessary steps).*

*a) Comment the code for:*

* + [*harmony light client*](https://github.com/harmony-one/horizon/blob/main/contracts/HarmonyLightClient.sol)
  + [*ethereum light client*](https://github.com/harmony-one/horizon/blob/main/contracts/EthereumLightClient.sol)
  + [*token locker on harmony*](https://github.com/harmony-one/horizon/blob/main/contracts/TokenLockerOnHarmony.sol)
  + [*token locker on ethereum*](https://github.com/harmony-one/horizon/blob/main/contracts/TokenLockerOnEthereum.sol)
  + [*test contract*](https://github.com/harmony-one/horizon/blob/main/test/bridge.hmy.js)

*Provide commented code in your submission.*

**[ANSWER]**

*b) Why HarmonyLightClient has* ***bytes32 mmrRoot*** *field and EthereumLightClient does not? (You will need to think of blockchain architecture to answer this)*

**[ANSWER]**

1. ***[Bonus]*** *What are checkpoint blocks? Do they differ from epoch blocks and how? Why are they used?*

**[ANSWER]**

1. **[Infrastructure only]** *Horizon still doesn’t use zk-proofs in order to speed up light clients. What changes would you need to make to the code in order to apply initial state sync through zk-snarks? Provide pseudo code of improved version of light client.*

**[ANSWER]**

### ****Rainbow Bridge****

1. ***[Infrastructure only]*** *Comment code implemented in* [*NearBridge.sol*](https://github.com/aurora-is-near/rainbow-bridge/blob/master/contracts/eth/nearbridge/contracts/NearBridge.sol)*. Note that they're using fraud proofs and not zk proofs.*

**[ANSWER]**

1. *Explain the differences between Rainbow bridge and Horizon bridge. Which approach would you take when building your own bridge (describe technology stack you would use)?*

**[ANSWER]**

1. ***[Bonus]*** *Explain how merkle mountain ranges work and how they can be used in order to do block inclusion proofs. (You can check FlyClient for a light client implementation that uses MMR).*

**[ANSWER]**

### ****Thinking In ZK****

1. *If you have a chance to meet with the people who built the above protocols what questions would you ask them?*

**[ANSWER]**